CHAPTER 19

Configuring a Windows File Sharing (Samba) Server

IN THIS CHAPTER

Getting and installing Samba

Using Samba security features

Editing the `smb.conf` configuration file

Accessing Samba from Linux and Windows clients

Using Samba in the enterprise

Samba is the project that implements open source versions of protocols used to share files and printers among Windows systems as well as authenticate users and restrict hosts. Samba offers a number of ways to share files among Windows, Linux, and MacOS systems that are well known and readily available to users of those systems.

This chapter steps you through the process of installing and configuring a Samba server. It describes the security features that you need to know to share your file and printer resources and describes how to access those resources from Linux and Windows systems.

Understanding Samba

Samba (`https://samba.org`) is a suite of programs that allows Linux, UNIX, and other systems to interoperate with Microsoft Windows file and printer sharing protocols. Windows, MacOS, and other client systems can access Samba servers to share files and printers in the same ways that they would from Windows file and print servers.

With Samba, you can use standard TCP/IP networking to communicate with clients. For name service, Samba supports regular TCP/IP hostnames as well as NetBIOS names. For that reason, Samba doesn't require the NetBEUI (Microsoft Raw NetBIOS frame) protocol. File sharing is done using Server Message Block (SMB) protocol, which is sometimes referred to as the *Common Internet File System (CIFS)*.

The Samba project has gone to great lengths to make its software secure and robust. In fact, many people prefer using Samba servers over Windows file servers because of the added security that is inherent in running Windows-style file sharing services on Linux or other UNIX-like operating systems.

Linux® Bible, Tenth Edition. Christopher Negus.

Beyond all of the technical mumbo-jumbo, however, the end result is that Samba makes it easy to share files and printers between Linux servers and Windows desktop systems. For the server, only a few configuration files and tools are needed to manage Samba. For the clients, shared resources just show up under Network selection in the File Explorer (formerly Windows Explorer) application or in the Network Neighborhood on older Windows systems.

To configure the Samba service, you directly edit Samba configuration files (particularly `smb.conf`) and run a few commands. Graphical and web-based interfaces, such as `system-config-samba` and Samba SWAT, are no longer included with the latest Fedora and RHEL systems.

To begin using Samba on your Linux system, you need to install a few software packages, as described in the next section.

Installing Samba

In Red Hat Enterprise Linux and Fedora, to configure a Samba file and print server, installing the `samba` package gets you everything you need to start. Among other components, the `samba` package includes the Samba service daemon (`/usr/sbin/smbd`) and NetBIOS name server daemon (`/usr/sbin/nmbd`). Installing the `samba` package pulls in the `samba-common` package, which contains server configuration files (`smb.conf`, `lmhosts`, and others) and commands for adding passwords and testing configuration files, along with other Samba features.

Features from other packages are referenced in this chapter, so I describe how to install those packages as well. Those packages include the following:

- **`samba-client` package**: Contains command-line tools such as `smbclient` (for connecting to Samba or Windows shares), `nmblookup` (for looking up host addresses), and `findsmb` (to find SMB hosts on the network).
- **`samba-winbind` package**: Includes components that allow your Samba server in Linux to become a complete member of a Windows domain, including using Windows user and group accounts in Linux.

To install all the packages just mentioned (`samba-common` is installed as a dependency of `samba`, so it doesn't need to be noted specifically), enter the following as root from the command line in Fedora or RHEL:

```
# yum install samba samba-client samba-winbind
...
Last metadata expiration check: 0:01:44 ago on Sun 24 Jan 2020
11:35:37 AM EST.
Dependencies resolved.
```

```
================================================================================
 Package       Architecture     Version  Repository
Size
================================================================================
Installing:
 samba         x86_64  4.10.4-101.el8_1  rhel-8-for-x86_64-baseos-
rpms 739 k
 samba-winbind x86_64  4.10.4-101.el8_1  rhel-8-for-x86_64-baseos-
rpms 570 k
Installing dependencies:
 samba-common-tools
               x86_64  4.10.4-101.el8_1  rhel-8-for-x86_64-baseos-
rpms 469 k
 samba-libs    x86_64  4.10.4-101.el8_1  rhel-8-for-x86_64-baseos-
rpms 185 k
 samba-winbind-modules
               x86_64  4.10.4-101.el8_1  rhel-8-for-x86_64-baseos-
rpms 122 k
 samba-client  x86_64  4.10.4-101.el8_1  rhel-8-for-x86_64-baseos-
rpms 658 k

Transaction Summary
================================================================================
Install  6 Packages

Total download size: 2.5 M
Installed size: 6.8 M
Is this ok [y/d/N]: y
```

After you have installed the Samba packages, look at the configuration files in the `samba-common` package:

```
# rpm -qc samba-common
/etc/logrotate.d/samba
/etc/samba/lmhosts
/etc/samba/smb.conf
/etc/sysconfig/samba
```

The `/etc/logrotate.d/samba` and `/etc/sysconfig/samba` files are usually not modified. The first sets how files in `/var/log/samba` log files are rotated (copied to other files and removed) over time. The second is a file where you could put options that are passed to the `smbd`, `nmbd`, or `winbindd` daemon, so you could turn off features such as debugging.

Most configuration files that you would modify for Samba are in the `/etc/samba` directory. The `smb.conf` file is the primary configuration file where you put global settings for the Samba server as well as individual file and printer share information (more on that later). The `lmhosts` file enables the Samba NetBIOS hostname to be mapped into IP addresses.

Although it doesn't exist by default, you can create a file named /etc/samba/smbusers to map Linux usernames into Windows usernames. As you configure your Samba server, you can refer to the smb.conf man page (man smb.conf). There are also man pages for Samba commands, such as smbpasswd (to change passwords), smbclient (to connect to a Samba server), and nmblookup (to look up NetBIOS information).

After you have installed Samba packages and completed a quick survey of what they contain, try starting up the Samba service and see what you get in a default configuration.

Starting and Stopping Samba

With samba and samba-common installed, you can start the server and investigate how it runs in the default configuration. Two main services are associated with a Samba server, each of which has its own service daemon:

smb: This service controls the smbd daemon process, which provides the file and print sharing services that can be accessed by Windows clients.

nmb: This service controls the nmbd daemon. By providing NetBIOS name service name-to-address mapping, nmbd can map requests from Windows clients for NetBIOS names so that they can be resolved into IP addresses.

To share files and printers with other Linux systems with Samba, only the smb service is required. The next section describes how to start and enable the smb service.

Starting the Samba (smb) service

The smb service is what starts the smbd server and makes files and printers available from your local system to other computers on the network. As usual, services are enabled and started differently on different Linux systems. For different Linux systems, you need to find the name of the service and the correct tool to start the smbd daemon.

In Fedora and RHEL, to enable Samba to start immediately when the system boots, enter the following from the command line as root:

```
# systemctl enable smb.service
# systemctl start smb.service
# systemctl status smb.service
smb.service - Samba SMB Daemon
  Loaded: loaded (/usr/lib/systemd/system/smb.service; enabled)
  Active: active (running) since Fri 2020-01-31 07:23:37 EDT; 6s ago
     Docs: man:smbd(8)
           man:samba(7)
           man:smb.conf(5)
   Status: "smbd: ready to serve connections..."
    Tasks: 4 (limit: 12216)
   Memory: 20.7M
 Main PID: 4838 (smbd)
CGroup: /system.slice/smb.service
```

```
   ├ 4838 /usr/sbin/smbd --foreground --no-process-group
   └ 4840 /usr/sbin/smbd --foreground --no-process-group
```

The first `systemctl` command enables the service, the second starts it immediately, and the third shows the status. To investigate further, notice that the service file is located at `/usr/lib/systemd/system/smb.service`. Look at the contents of that file:

```
# cat /usr/lib/systemd/system/smb.service
[Unit]
Description=Samba SMB Daemon
Documentation=man:smbd(8) man:samba(7) man:smb.conf(5)
Wants=network-online.target
After=network.target network-online.target nmb.service winbind.
service
[Service]
Type=notify
NotifyAccess=all
PIDFile=/run/smbd.pid
LimitNOFILE=16384
EnvironmentFile=-/etc/sysconfig/samba
ExecStart=/usr/sbin/smbd --foreground --no-process-group $SMBDOPTIONS
ExecReload=/bin/kill -HUP $MAINPID
LimitCORE=infinity
Environment=KRB5CCNAME=FILE:/run/samba/krb5cc_samba
[Install]
WantedBy=multi-user.target
```

The Samba daemon process (`smbd`) starts up after the `network`, `network-online`, `nmb`, and `winbind` targets. The `/etc/sysconfig/samba` file contains variables that are passed as arguments to the `smbd`, `nmbd`, and `winbindd` daemons when they start. No options are set by default for any of those daemons. The `WantedBy` line indicates that `smb.service` should start when the system boots up into multi-user mode (`multi-user.target`), which it does by default.

In RHEL 6 and earlier, you can start the Samba service as follows:

```
# service smb start
Starting SMB services:        [  OK  ]
# chkconfig smb on
# service smb status
smbd (pid  28056) is running...
# chkconfig --list smb
smb             0:off  1:off  2:on  3:on  4:on  5:on  6:off
```

Whether you are running your Samba server on RHEL, Fedora, or another Linux system, you can check access to the Samba server using the `smbclient` command (from the `samba-client` package). You can get basic information from a Samba server using the following command:

```
# smbclient -L localhost
Enter SAMBA\root's password: <ENTER>
```

```
Anonymous login successful

   Sharename  Type    Comment
   ---------  ----    -------
   print$     Disk    Printer Drivers
   IPC$       IPC     IPC Service
  (Samba Server Version 4.10.10)
   deskjet    Printer deskjet
Reconnecting with SMB1 for workgroup listing.
Anonymous login successful
    Server               Comment
    ---------            -------
    Workgroup            Master
    ---------            -------
```

The `smbclient` output allows you to see what services are available from the server. By default, anonymous login is allowed when querying the server (so I just pressed Enter when prompted for a password).

You can discern a number of things about the default Samba server setup from this output:

- All printers that are shared via the CUPS server on your Linux system are, by default, also made available from the Samba server running on that same system.
- No directories are shared yet from the server.
- There is no NetBIOS name service running yet from the Samba server.

Next, you can decide whether you want to run the NetBIOS name service on your Samba server.

Starting the NetBIOS (nmbd) name server

If no Windows domain server is running on the network, as is the case here, you can start the `nmb` service on the Samba host to provide that service. To start the `nmb` service (`nmbd` daemon) in Fedora or RHEL 7, type the following:

```
# systemctl enable nmb.service
# systemctl start nmb.service
# systemctl status nmb.service
```

In RHEL 6 and earlier, you would type the following to start the `nmb` service:

```
# service nmb start
# service nmb status
# chkconfig nmb on
# chkconfig --list nmb
```

Regardless of how the NetBIOS service was started, the `nmbd` daemon should now be running and ready to serve NetBIOS name-to-address mapping. Run the `smbclient -L` command again, followed by the IP address of the server. This time, the last few lines of

the output should show the information obtained from the NetBIOS server now running on the Samba server. In this case, the last few lines look like this:

```
# smbclient -L localhost
    ...
    Workgroup   Master
    ---------   -------
    SAMBA       FEDORA30
```

You can see that the new NetBIOS server's name is `FEDORA30` and that it is the master server for the workgroup. To query the `nmbd` server for the IP address of `FEDORA30`, you would enter the following:

```
# nmblookup -U localhost FEDORA30
querying FEDORA30 on 127.0.0.1
192.168.122.81 FEDORA30<00>
```

You should be able to see your Samba server running from the local system now. The hostname assigned to the system (in this case `FEDORA30`) is assigned by default.

However, if you have a firewall configured or SELinux enabled, you may not be able to access the Samba server fully from a remote system yet. The next section should help you to open Samba to systems outside of the local system as well as to allow some Samba features that may be turned off by SELinux.

Stopping the Samba (smb) and NetBIOS (nmb) services

To stop the `smb` and `nmb` services in Fedora or RHEL, you can use the same `systemctl` command that you used to start them. You can use the same command to disable the services as well so that they do not start up again when the system boots. Here are examples of how to stop the `smb` and `nmb` services immediately:

```
# systemctl stop smb.service
# systemctl stop nmb.service
```

In RHEL 6 and earlier, you would enter the following to stop the `smb` and `nmb` services:

```
# service smb stop
# service nmb stop
```

To prevent the `smb` and `nmb` services from starting the next time the system reboots, enter the following commands in Fedora or RHEL:

```
# systemctl disable smb.service
# systemctl disable nmb.service
```

In Red Hat Enterprise Linux 6 and earlier, enter the following commands to disable the `smb` and `nmb` services:

```
# chkconfig smb off
# chkconfig nmb off
```

Of course, you only want to stop or disable the `smb` and `nmb` services if you no longer want to use the Samba service. If you are ready to continue to configure your Samba service, you can continue on and begin to configure your Linux security features to allow the Samba service to become available to others on your network.

Securing Samba

If you cannot access your Samba server immediately after starting it, you probably have some security work to do. Because many default installations of Linux prevent, rather than allow, access to the system, dealing with security for a service such as Samba usually has more to do with making it available than making it secure.

Here are the security features that you should be aware of when configuring your Samba system:

Firewalls The default firewall for Fedora, RHEL, and other Linux systems prevents any access to local services from outside systems. So, to allow users from other computers to access your Samba service, you must create firewall rules that open one or more ports for selected protocols (TCP in particular).

SELinux Many features of Samba are designated as potentially insecure by SELinux. Because the default SELinux Booleans (on/off switches for certain features) are set to provide the least access required, you need to turn Booleans on for features such as allowing users to access their own home directories with Samba. In other words, you can configure Samba to share user home directories, but SELinux prohibits someone from trying to use that feature unless you explicitly configure SELinux to allow that feature.

Host and user restrictions Within the Samba configuration files themselves, you can indicate which hosts and users can have access to the Samba server as a whole or to particular shared directories.

The next sections describe how to set up the security features just mentioned for Samba.

Configuring firewalls for Samba

If an `iptables` or `firewalld` firewall is configured for your system when you first install it, the firewall typically allows any requests for services from local users but none by outside users. That's why, at the end of the installation section of this chapter, you should have been able to test that Samba was working using the `smbclient` command from the local system. However, if the request originated from another system, it would have been rejected.

Configuring firewall rules for Samba mainly consists of opening up incoming ports on which the `smbd` and `nmbd` daemons are listening. These are the ports that you should open to get a working Samba service on your Linux system:

TCP port 445: This is the primary port on which the Samba `smbd` daemon listens. Your firewall must support incoming packet requests on this port for Samba to work.

TCP port 139: The `smbd` daemon also listens on TCP port 139 in order to handle sessions associated with NetBIOS hostnames. It is possible to use Samba over TCP without opening this port, but it is not recommended.

UDP ports 137 and 138: The `nmbd` daemon uses these two ports for incoming NetBIOS requests. If you are using the `nmbd` daemon, these two ports must be open for new packet requests for NetBIOS name resolution.

For Fedora and RHEL, allowing incoming access to those four ports is easy. Simply open the Firewall Configuration window, and select the check boxes next to the `samba` and `samba-client` entries on the public zone, Services tab. Those ports become immediately accessible (no restart of the `firewalld` service is required).

For earlier Fedora and RHEL systems that use `iptables` directly instead of the `firewalld` service, opening the firewall is a more manual process. Consider a default firewall from Fedora that allows incoming packets from the local host, from established connections, and related to established connections but denies all other incoming packets. The following example represents a set of firewall rules in the `/etc/sysconfig/iptables` file, with four new rules (highlighted in the example that follows) added to open ports for Samba:

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-I INPUT -m state --state NEW -m udp -p udp --dport 137 -j ACCEPT
-I INPUT -m state --state NEW -m udp -p udp --dport 138 -j ACCEPT
-I INPUT -m state --state NEW -m tcp -p tcp --dport 139 -j ACCEPT
-I INPUT -m state --state NEW -m tcp -p tcp --dport 445 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

Your firewall may include additional rules to allow incoming packet requests for other services, such as Secure Shell (`sshd`) or web (`httpd`) services. You can leave those in place. The main point is to have your Samba rules placed somewhere before the final `REJECT` rules.

If your `iptables` firewall is enabled, you can restart it to have the new rules take effect. To do that, type **`systemctl restart iptables.service`** (in older Fedora systems) or **`service restart iptables`** (in RHEL 6 or earlier). Try connecting to the Samba service by using the `smbclient` command again, or by using other techniques described in the section "Accessing Samba Shares" later in this chapter.

See Chapter 25, "Securing Linux on a Network," for more information on using `iptables`.

Configuring SELinux for Samba

There are both file context and Boolean considerations related to using Samba with SELinux in enforcing mode. File contexts must be properly set on a directory that is shared by Samba. Booleans allow you to override the secure-by-default approach to certain Samba features.

You can find information on how SELinux confines Samba on the `samba_selinux` man page (`man samba_selinux`). You must install the `selinux-policy-doc` package to get that man page. For a deeper understanding of SELinux, refer to Chapter 24, "Enhancing Linux Security with SELinux."

Setting SELinux Booleans for Samba

An easy way to list and change SELinux Booleans for Samba is from the command line. To use the `semanage` command to list Samba-related Booleans, enter the following:

```
# semanage boolean -l | egrep "smb|samba"
```

The following is a list of SELinux Booleans that apply to Samba and their descriptions. Most of the Booleans let you set which files and directories the Samba server can read and write on behalf of Samba users. Others let you allow potentially insecure features:

samba_run_unconfined: Allows samba to run unconfined scripts from Samba shares.

smbd_anon_write: Allows Samba to let anonymous users modify public files used for public file transfer services. Files and directories must be labeled `public_content_rw_t`.

samba_enable_home_dirs: Allows Samba to share users' home directories.

samba_export_all_ro: Allows Samba to share any file and directory read-only.

use_samba_home_dirs: Allows a remote Samba server to access home directories on the local machine.

samba_create_home_dirs: Allows Samba to create new home directories (for example, via PAM).

samba_export_all_rw: Allows Samba to share any file or directory read/write.

The following Booleans affect Samba's ability to share directories that are themselves mounted from other remote services (such as NFS) or to act as a Windows domain controller:

samba_share_fusefs: Allows Samba to export `ntfs/fusefs` volumes.

samba_share_nfs: Allows Samba to export NFS volumes.

samba_domain_controller: Allows Samba to act as the domain controller, add users and groups, and change passwords.

The `setsebool` command is used to turn the SELinux Booleans on or off. Used with the `-P` option, `setsebool` sets the Boolean you indicate permanently. For example, to allow

Samba to share any file or directory with read-only permission from the server, you could type the following from a shell as root user:

```
# setsebool -P samba_export_all_ro on
# getsebool samba_export_all_ro
samba_export_all_ro --> on
```

The `setsebool` command sets the Boolean in this case to `on`. The `getsebool` lets you see the value of the Boolean.

Setting SELinux file contexts for Samba

SELinux confines the files that the Samba service can access. Instead of allowing any file with the proper read and write permission to be shared by the Samba server, SELinux (when in enforcing mode) requires that files and directories have the correct file contexts set on them before the Samba service can even see that the files exist.

In order for the Samba service to function with SELinux immediately, some files and directories come preset with the proper file contexts. For example, Samba configuration files (`/etc/samba/*`), log files (`/var/log/samba/*`), and libraries (`/var/lib/samba/*`) have rules assigned to ensure that they get the proper file contexts. To find files and directories associated with the Samba service and `smbd` daemon that have file contexts preset, run the following:

```
# semanage fcontext -l | grep -i samba
# semanage fcontext -l | grep -i smb
```

The file context portion in which you are interested ends with `_t`: for example, `samba_etc_t`, `samba_log_t`, and `samba_var_t` for the `/etc/samba`, `/var/log/samba`, and `/var/lib/samba` directories, respectively.

You may find that you need to change file contexts—for example, when you put files in nonstandard locations (such as moving the `smb.conf` file to `/root/smb.conf`) or when you want to share a directory (other than home directories, which can be turned on by setting a Boolean). Unlike the `vsftpd` (FTP) and `httpd` (web) servers that come with Linux, Samba has no default shared content directories (those just mentioned used `/var/ftp` and `/var/www/html`).

You can change a file context permanently by creating a new file context rule and then applying that rule to the file or directory for which it is intended. You can do that with the `semanage` command (to make the rule) and `restorecon` command (to apply the rule). For example, if you wanted to share a directory, `/mystuff`, you would create that directory with the proper permissions and run the following command to make it available for read/write access from Samba:

```
# semanage fcontext -a -t samba_share_t "/mystuff(/.*)?"
# restorecon -v /mystuff
```

After those commands are run, the `/mystuff` directory, along with any files and directories below that point, have the file context of `samba_share_t`. It is then up to you

to assign the correct Linux ownership and file permissions to allow access to the users you choose. The upcoming section "Configuring Samba" provides an example of creating a share, and it shows you how to add permissions and ownership to a shared directory using standard Linux commands.

Configuring Samba host/user permissions

Within the `smb.conf` file itself, you can allow or restrict access to the entire Samba server or to specific shares based on the hosts or users trying to gain access. You can also restrict access to the Samba server by providing the service only on particular interfaces.

For example, if you have one network interface card connected to the Internet and another connected to the local network, you can tell Samba to serve requests only on the local network interface. The next section describes how to configure Samba, including how to identify which hosts, users, or network interfaces can access your Samba server.

Configuring Samba

Inside the `/etc/samba/smb.conf` file are settings for configuring your Samba server, defining shared printers, configuring how authentication is done, and creating shared directories. The file consists of the following predefined sections:

`[global]` Settings that apply to the Samba server as a whole are placed in this section. This is where you set the server's description, its workgroup (domain), the location of log files, the default type of security, and other settings.

`[homes]` This section determines whether users with accounts on the Samba server can see their home directories (browseable) or write to them.

`[printers]` In this section, settings tell Samba whether to make printers available through Samba that are configured for Linux printing (CUPS).

`[print$]` This section configures a directory as a shared printer drivers folder.

Inside the `smb.conf` file, lines beginning with pound signs (#) or semicolons (;) are comments. Removing the semicolons enables you to set up different kinds of shared information quickly. The # sign can also be used to comment out a line.

When you begin editing your `smb.conf` file, make a backup that you can go back to if something goes wrong. You can start by copying the `smb.conf.example` file to `smb.conf`, if you want to start with some examples.

Configuring the [global] section

Here is an example of a `[global]` section of the `smb.conf` file:

```
[global]
        workgroup = SAMBA
        security = user
```

```
        passdb backend = tdbsam
        printing = cups
        printcap name = cups
        load printers = yes
        cups options = raw

;       netbios name = MYSERVER
;       interfaces = lo eth0 192.168.12.2/24 192.168.13.2/24
;       hosts allow = 127. 192.168.12. 192.168.13.
```

The `workgroup` (also used as the domain name) is set to `SAMBA` in this example. When a client communicates with the Samba server, this name tells the client which workgroup the Samba server is in.

The default `security` type is set to `user` (Samba usernames and passwords).

The `passdb backend = tdbsam` specifies to use a Samba backend database to hold passwords. You can use the `smbpasswd` command to set each user's password (as described later).

Setting `printing = cups` and `printcap name = cups` indicates to use the `printcap` created by the CUPS printing service. When you set `load printers = yes`, Samba knows to share any printers configured by your local CUPS printing service from Samba.

The `cups options` lets you pass any options that you like to the CUPS printers served by your Samba server. By default, only `raw` is set, which allows Windows clients to use their own print drivers. Printers on your Samba server print the pages they are presented in raw form.

By default, your server's DNS hostname (enter `hostname` to see what it is) is used as your Samba server's NetBIOS name as well. You can override that and set a separate NetBIOS name by uncommenting the `netbios name` line and adding the server name you want. For example, `netbios name = myownhost`. `localhost` is used as your NetBIOS name if it has not otherwise been set.

If you want to restrict access to the Samba server so that it only responds on certain interfaces, you can uncomment the `interfaces` line and add either the IP address or name (`lo`, `eth0`, `eth1`, and so on) of the network interfaces you want.

You can restrict access to the Samba server to specific hosts as well. Uncomment the `hosts allow` line (remove the semicolon) and insert the IP addresses of the hosts that you want to allow. To enter a range of addresses, simply end the subnetwork portion of the address, followed by a dot. For example, `127.` is associated with IP addresses that point to the local host. The 192.168.12. entry matches all IP addresses from 192.168.12.1 to 192.168.12.254.

Configuring the [homes] section

The `[homes]` section is configured, by default, to allow any Samba user account to be able to access its own home directory via the Samba server. Here is what the default homes entry looks like:

```
[homes]
        comment = Home Directories
        valid users = %S, %D%w%S
        browseable = No
        read only = No
        inherit acls = Yes
```

Setting `valid users` to `%S` substitutes the current service name, which allows any valid users of the service to access their home directories. The valid users are also identified by domain or workgroup (`%D`), winbind separator (`%w`), and name of current service (`%S`).

The `browseable = No` setting prevents the Samba server from displaying the availability of the shared home directories. Users who can provide their own Samba usernames and passwords can read and write in their own home directories (`read only = no`). With `inherit acls` set to `Yes`, access control lists can be inherited to add another layer of security on the shared files.

If after starting the `smb` service you cannot log in using a valid user account, you may need to change some security features on your system. On Fedora and RHEL systems, in particular, SELinux features need to be changed to allow users to access their home directories if you are in SELinux enforcing mode.

For example, if you tried to use `smbclient` to log in to your home directory, the login would succeed, but when you tried to list the contents of the home directory, you might see the following message:

```
NT_STATUS_ACCESS_DENIED listing \*
```

To tell SELinux to allow Samba users to access their home directories as Samba shares, turn on the `samba _ enable _ home _ dirs` Boolean by entering the following as root from a shell:

```
# setsebool -P samba_enable_home_dirs on
```

The `setsebool` command turns on the capability of Samba to share home directories (which is off by default). First create a password for the user with `smbpasswd` and then log in with `smbclient`. The form for using the `smbclient` command to check access to the user's home directory, again for the user `chris`, would be the following (replacing the IP address with the name or address of your Samba server):

```
$ smbpasswd -a chris
New SMB password: *********
Retype new SMB password: *********
Added user chris.

$ smbclient -U chris //192.168.0.119/chris

Enter SAMBA\chris's password:
Try "help" to get a list of possible commands.
smb: \> ls file.txt
  file.txt 149946368  Sun Jan  4 09:28:53 2020
          39941 blocks of size 524288. 28191 blocks available
smb:\>  quit
```

The main point to remember is that, even though the share is not browseable, you can request it by giving the Samba server's hostname or IP address, followed by the user's name (here, `chris`), to access the user's home directory.

Configuring the [printers] section

Any printer that you configure for CUPS printing on your Linux system is automatically shared to others over Samba, based on the `[printers]` section that is added by default. The global `cups options = raw` setting makes all printers raw printers (meaning that the Windows client needs to provide the proper printer driver for each shared printer).

Here's what the default printers section looks like in the `smb.conf` file:

```
[printers]
        comment = All Printers
        path = /var/tmp
        printable = Yes
        create mask = 0600
        browseable = No
```

The `path` tells Samba to store temporary print files in `/var/tmp`. The `printable = Yes` line causes all of your CUPS printers on the local system to be shared by Samba. Printers are writeable and allow guest printing by default. The `create mask = 0600` setting used here has the effect of removing write and execute bits for group and other, within the ACL, when files are created in the path directory.

To see that local printers are available, you could run the `smbclient -L` command from a Linux system, as shown earlier. On a Windows system, you can select Network from the File Explorer window and select the icon representing your Samba server. All shared printers and folders appear in that window. (See the section "Accessing Samba Shares" later in this chapter for details on viewing and using shared printers.)

Creating a Samba shared folder

Before you can create a shared folder, that folder (directory) must exist and have the proper permissions set. In this example, the `/var/salesdata` directory is shared. You want the data to be writeable by the user named `chris` but visible to anyone on your network. To create that directory and set the proper permissions and SELinux file contexts, type the following as root user:

```
# mkdir /var/salesdata
# chmod 775 /var/salesdata
# chown chris:chris /var/salesdata
# semanage fcontext -a -t samba_share_t /var/salesdata
# restorecon -v /var/salesdata
# touch /var/salesdata/test
# ls -lZ /var/salesdata/test
-rw-r--r--. 1 root root
    unconfined_u:object_r:samba_share_t:s0 0 Dec 24 14:35
        /var/salesdata/test
```

Adding the shared folder to Samba

With the /var/salesdata directory created and properly configured to be shared by Samba, here is what the shared folder (called salesdata) might look like in the smb.conf file:

```
[salesdata]
        comment = Sales data for current year
        path = /var/salesdata
        read only = no
;       browseable = yes
        valid users = chris
```

Before this share was created, the /var/salesdata directory was created, with chris assigned as the user and group, and the directory was set to be readable and writeable by chris. (The SELinux file context must also be set if SELinux is in enforcing mode.) The Samba username chris must be presented along with the associated password to access the share. After chris is connected to the share, chris has read and write access to it (read only = no).

Now that you have seen the default settings for Samba and an example of a simple shared directory (folder), read the next few sections to see how to configure shares even further. In particular, the examples demonstrate how to make shares available to particular users, hosts, and network interfaces.

Checking the Samba share

For the changes to your Samba configuration to take effect, you need to restart the smb service. After that is done, check that the Samba share you created is available and that any user you assigned to the share can access it. To do those things, enter the following as root user from a shell on the Samba server:

```
# systemctl restart smb.service
# smbclient -L localhost -U chris
Enter SAMBA\chris's password: *******
    Sharename       Type      Comment
    ---------       ----      -------
    salesdata       Disk      Sales data for current year
    print$          Disk      Printer Drivers
    IPC$            IPC       IPC Service (Samba 4.10.4)
    chris           Disk      Home Directories
Reconnecting with SMB1 for workgroup listing.
    Server               Comment
    ---------            -------

    Workgroup            Master
    ---------            -------
    SAMBA                FEDORA30
...
```

Here you can see the share name (salesdata), the domain set to the workgroup name SAMBA, and the description entered earlier (Sales data for current year). Next, a quick way to test access to the share is to use the smbclient command. You can use the hostname or IP address with smbclient to access the share. Because I am on the local system in this example, I just use the name localhost and the user I added (chris):

```
# smbclient -U chris //localhost/salesdata
Enter SAMBA\chris's password: ********
Try "help" to get a list of possible commands.
smb: \> lcd /etc
smb: \> put hosts
putting file hosts as \hosts (43.5 kb/s) (average 43.5 kb/s)
smb: \> ls
  .                                   D        0  Sun Dec 29 09:52:51 2020
  ..                                  D        0  Sun Dec 29 09:11:50 2020
  hosts                               A       89  Sun Dec 29 09:52:51 2020
          39941 blocks of size 524288. 28197 blocks available
smb: \> quit
```

A Samba share is in the form //host/share or \\host\share. However, when you identify a Samba share from a Linux shell in the latter case, the backslashes need to be escaped. So, as an argument, the first example of the share would have to appear as \\\\localhost\\salesdata. Thus, the first form is easier to use.

Note

Escaping a character that you type from the shell is done by putting a backslash (\) in front of that character. It tells the shell to use the character following the backslash literally, instead of giving the character a special meaning to the shell. (The * and ? characters are examples of characters with special meaning.) Because the backslash itself has special meaning to the shell, if you want to use a backslash literally, you need to precede it with a backslash. That is why when you want to type a Samba address that includes two backslashes, you actually have to enter four backslashes.

When prompted, enter the Samba password for that user (it may be different from the Linux user's password). The Samba user's password was set earlier with smbpasswd in this example. You see the smb: \> prompt after that.

At this point, you have a session open to the Samba host that is similar to an ftp session for traversing an FTP server. The lcd /etc command makes /etc the current directory on the local system. The put hosts command uploads the hosts file from the local system to the shared directory. Typing ls shows that the file exists on the server. The quit command ends the session.

Restricting Samba access by network interface

To restrict access to all of your shares, you can set the global interfaces setting in the smb.conf file. Samba is designed more for local file sharing than for sharing over wide area networks. If your computer has a network interface connected to a local network and one connected to the Internet, consider allowing access only to the local network.

To set which interfaces Samba listens on, uncomment the `interfaces` line shown in an earlier example in the `[global]` section of the `smb.conf` file. Then add the interface names or IP address ranges of those computers that you want to allow access to your computer. Here is an example:

```
interfaces = lo 192.168.22.15/24
```

This interfaces entry allows access to the Samba service to all users on the local system (`lo`). It also allows access to any systems on the 192.168.22 network. See the `smb.conf` man page's description of different ways of identifying hosts and network interfaces.

Restricting Samba access by host

Host access to the Samba server can be set for the entire service or for single shares.

Here are some examples of `hosts allow` and `hosts deny` entries:

```
hosts allow = 192.168.22. EXCEPT 192.168.22.99
hosts allow = 192.168.5.0/255.255.255.0
hosts allow = .example.com market.example.net
hosts deny = evil.example.org 192.168.99.
```

These entries can be put in the `[global]` section or in any shared directory section. The first example allows access to any host in the 192.168.22. network except for 192.168.22.99, which is denied. Note that a dot is required at the end of the network number. The 192.168.5.0/255.255.255.0 example uses netmask notation to identify 192.168.5 as the set of addresses that are allowed.

In the third line of the sample code, any host from the `.example.com` network is allowed, as is the individual host `market.example.net`. The `hosts deny` example shows that you can use the same form to identify names and IP addresses in order to prevent access from certain hosts.

Restricting Samba access by user

Particular Samba users and groups can be allowed access to specific Samba shares by identifying those users and groups within a share in the `smb.conf` file. Aside from guest users, which you may or may not allow, the default user authentication for Samba requires you to add a Samba (Windows) user account that maps into a local Linux user account.

To allow a user to access the Samba server, you need to create a password for the user. Here is an example of how to add a Samba password for the user `jim`:

```
# smbpasswd -a jim
New SMB password: *******
Retype new SMB password: *******
```

After running that `smbpasswd` command, `jim` can use that username and password to access the Samba server. The `/var/lib/samba/private/passdb.tdb` file holds the password just entered for `jim`. After that, the user `jim` can change the password by simply typing `smbpasswd` when he is logged in. The root user can change the password by rerunning the command shown in the example but dropping the `-a` option.

If you wanted to give `jim` access to a share, you could add a `valid users` line to that shared block in the `smb.conf` file. For example, to provide both `chris` and `jim` access to a share, you could add the following line:

```
valid users = jim, chris
```

If the `read only` option is set to `no` for the share, both users could potentially write files to the share (depending on file permissions). If `read only` is set to `yes`, you could still allow access to `jim` and `chris` to write files by adding a `write list` line as follows:

```
write list = jim, chris
```

The write list can contain groups (that is, Linux groups contained in the `/etc/group` file) to allow write permission to any Linux user that belongs to a particular Linux group. You can add write permission for a group by putting a plus (+) character in front of a name. For example, the following adds write access for the `market` group to the share with which this line is associated:

```
write list = jim, chris, +market
```

There are many ways to change and extend the features of your shared Samba resources. For further information on configuring Samba, be sure to examine the `smb.conf` file itself (which includes many useful comments) and the `smb.conf` man page.

Accessing Samba Shares

After you have created some shared directories in Samba, many client tools are available in both Linux and Windows for accessing those shares. Command-line tools in Linux include the `smbclient` command, demonstrated earlier in this chapter. For a graphical means of accessing shares, you can use the file managers available in both Windows (File Explorer) and Linux (Nautilus, with the GNOME desktop).

Accessing Samba shares in Linux

Once a Samba share is available, it can be accessed from remote Linux and Windows systems using file managers or remote mount commands.

Accessing Samba shares from a Linux file manager

Opening a file manager in Linux can provide you with access to the shared directories from Linux (Samba) and Windows (SMB). How you access the file manager is different on different Linux desktops. In GNOME 3, you can click the Files icon. In other desktops, open the Home folder.

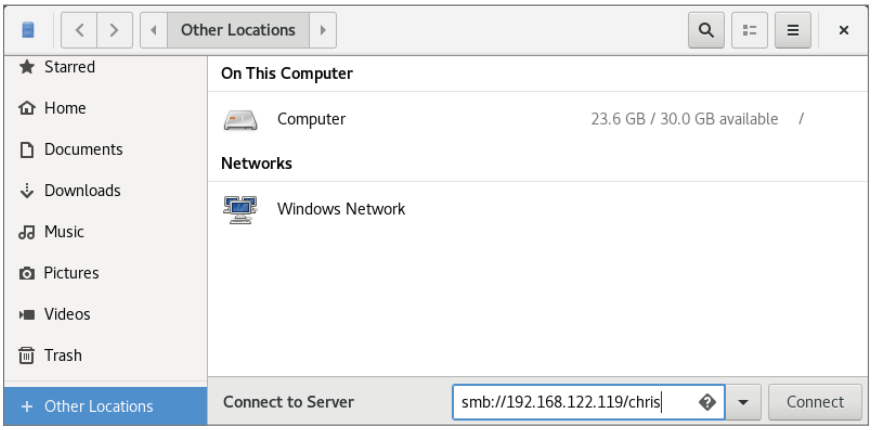
With the Nautilus window manager displayed, select Other Location in the left navigation bar. Available networks (such as Windows Network) should appear. Look to the box at the bottom of the window identified as Connect to Server, and then enter the location of an available Samba share. Given the previous examples, you would be able to use either of these shares:

```
smb://192.168.122.119/chris
smb://192.168.122.119/salesdata
```

The window should appear similar to Figure 19.1:

FIGURE 19.1

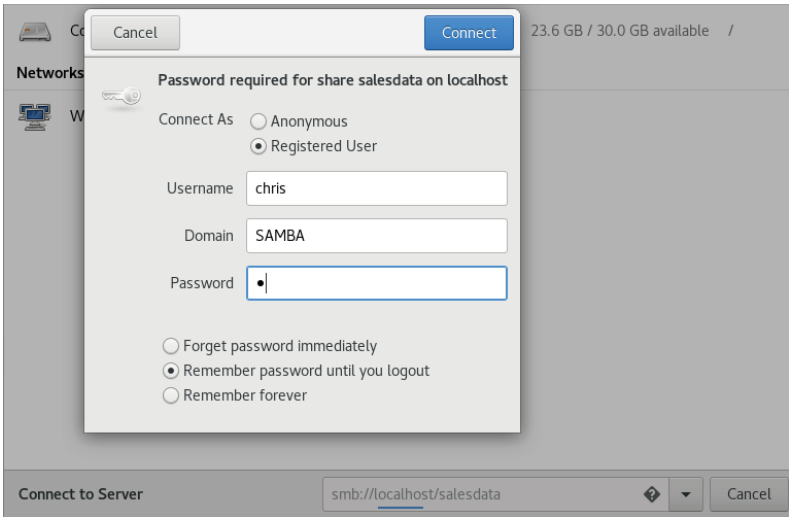
Identify a Samba share from the Nautilus Connect to Server box.



Click Connect. From the window that appears, you can select to connect as a registered user. If you do that, you can enter your username, Samba domain name, and the password for your user. You can also select whether or not to save that password. Figure 19.2 shows an example of that window:

FIGURE 19.2

Add your Samba credentials.

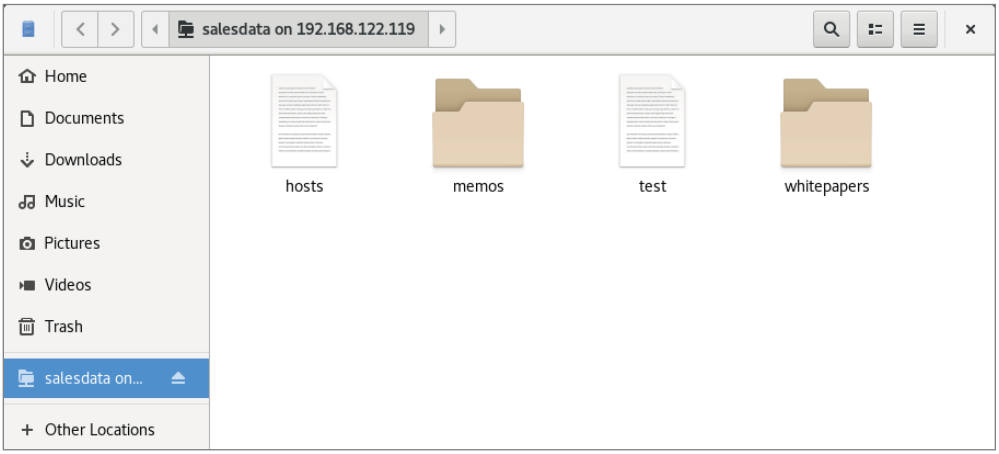


Click Connect.

If the user and password are accepted, you should see the contents of the remote directory. If you have write access to the share, you can open another Nautilus window and drag and drop files between the two systems. Figure 19.3 shows an example of the Nautilus window after I have connected to the `salesdata` share.

FIGURE 19.3

Displaying a Samba share from Connect to Server in Nautilus



Mounting a Samba share from a Linux command line

Because a Samba shared directory can be viewed as a remote filesystem, you can use common Linux tools to connect a Samba share (temporarily or permanently) to your Linux system. Using the standard `mount` command (with `cifs-utils` installed), you can mount a remote Samba share as a CIFS filesystem in Linux. This example mounts the `salesdata` share from the host at IP address 192.168.0.119 on the local directory `/mnt/sales`:

```
# yum install cifs-utils -y
# mkdir /mnt/sales
# mount -t cifs -o user=chris \
      //192.168.0.119/salesdata /mnt/sales
Password for chris@//192.168.122.119/salesdata: *******
# ls /mnt/sales
hosts   memos  test  whitepapers
```

When prompted, enter the Samba password for `chris`. Given that the user `chris` in this example has read-write permission to the shared directory, users on your system should be able to read and write to the mounted directory. Regardless of who saves files on the shared

directory, on the server those files are owned by the user `chris`. This mount lasts until the system is rebooted or you run the `umount` command on the directory. If you want the share to be mounted permanently (that is, every time the system boots up) in the same location, you can do some additional configuration. First, open the `/etc/fstab` file and add an entry similar to the following:

```
//192.168.0.119/salesdata /mnt/sales cifs credentials=/root/cif.txt 0 0
```

Next, create a credentials file (in this example, `/root/cif.txt`). In that file, put the name of the user and the user's password that you want to present when the system tries to mount the filesystem. Here is an example of the contents of that file:

```
user=chris
pass=mypass
```

Before you reboot to check that the entry is correct, try mounting it from the command line. A `mount -a` command tries to mount any filesystem listed in the `/etc/fstab` file that is not already mounted. The `df` command shows information about disk space for the mounted directory, as in the following example:

```
# mount -a
# df -h /mnt/sales
Filesystem                   Size  Used  Avail   Ues%  Mounted on
//192.168.0.119/salesdata     20G  5.7G    14G    30%  /mnt/sales
```

You should now be able to use the shared Samba directory as you do any directory on the local system.

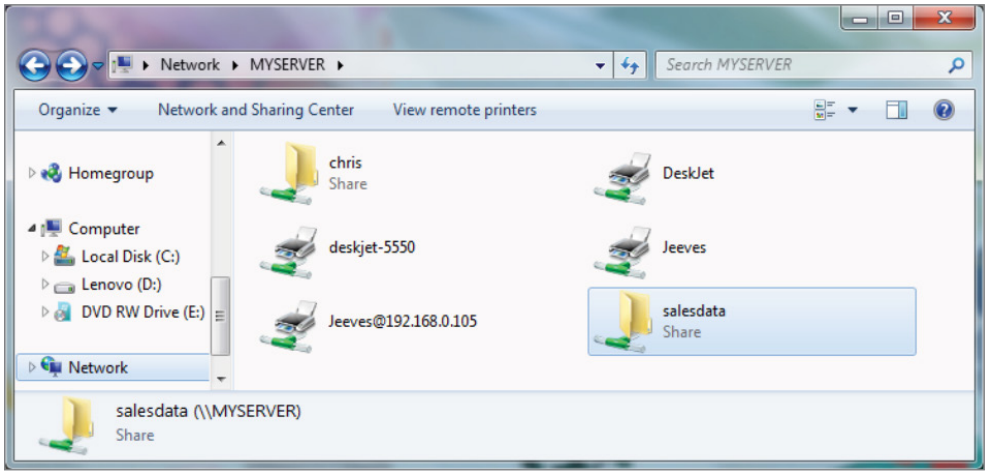
Accessing Samba shares in Windows

As with Linux, you can access Samba shares from the file manager window, in this case Windows File Explorer. To do this, open any folder in Windows, and select Network from the left panel. An icon representing the Samba server should appear on the screen. Click that icon and enter a password if prompted for one. You should see all shared printers and folders from that server (see Figure 19.4).

In Figure 19.4, you can see that there are two shared folders (directories): `chris` and `salesdata`. There are also several shared printers. To use the folders, double-click them and enter the required authentication information. Because printers are set up to use raw drivers by default, you need to obtain Windows drivers to use any of the Samba printers.

FIGURE 19.4

Accessing Samba shares from Windows



Using Samba in the Enterprise

Although it's beyond the scope of this book, Windows file and printer sharing via Samba servers is a very popular application in large enterprises. Despite the fact that Linux has begun to dominate the enterprise-quality server market, Microsoft Windows systems are still the predominant systems used on the desktop.

The major features needed to integrate Samba servers into a large enterprise with many Microsoft Windows desktops are related to authentication. Most large enterprises use Microsoft Active Directory Services (ADS) servers for authentication. On the Linux side, that means configuring Kerberos on the Linux system and using ADS (instead of user) for the type of security in the `smb.conf` file.

The advantage of central authentication is that users have to remember only one set of credentials throughout the enterprise and system administrators need to manage fewer user accounts and passwords.

Summary

Because of the popularity of Windows desktops, Samba servers have become popular for sharing files and printers among Windows and Linux systems. Samba provides a way to interoperate with Windows systems by implementing the Server Message Block (SMB) or Common Internet File (CIFS) protocol for sharing resources over a network.

This chapter stepped through the process of installing, starting, securing, configuring, and accessing Samba servers on a Linux system. Using command-line tools, I demonstrated how to set up a Samba server. I showed you both command-line and desktop tools for getting to Samba shares from Linux and Windows systems.

The next chapter describes the Network File System (NFS) facility. NFS is the native Linux facility for sharing and mounting filesystems over networks with other Linux and UNIX systems.

Exercises

The exercises in this section describe tasks related to setting up a Samba server in Linux and accessing that server using a Samba client. As usual, there are often several ways to accomplish some of the tasks here. So don't worry if you don't go about the exercises in exactly the same way as shown in the answers, as long as you get the same results. See Appendix B for suggested solutions.

Don't do these exercises on a Linux system running a Samba server because they will almost certainly interfere with that server. These exercises were tested on a Fedora system. Some of the steps might be slightly different on another Linux system.

1. Install the `samba` and `samba-client` packages.
2. Start and enable the `smb` and `nmb` services.
3. Set the Samba server's workgroup to `TESTGROUP`, the `netbios name` to `MYTEST`, and the server string to `Samba Test System`.
4. Add a Linux user named `phil` to your system, and add a Linux password and Samba password for `phil`.
5. Set the `[homes]` section so that home directories are browseable (`yes`) and writeable (`yes`), and `phil` is the only valid user.
6. Set any SELinux Boolean that is necessary to make it so that `phil` can access his home directory via a Samba client, then restart the `smb` and `nmb` services.
7. From the local system, use the `smbclient` command to list that the `homes` share is available.
8. From a Nautilus (file manager) window on the local system, connect to the `homes` share for the user `phil` on the local Samba server in a way that allows you to drag and drop files to that folder.
9. Open up the firewall so that anyone who has access to the server can access the Samba service (`smbd` and `nmbd` daemons).
10. From another system on your network (Windows or Linux), try to open the `homes` share again as the user `phil`, and again make sure that you can drag and drop files to it.